Webサイト評価のための多端末シミュレータ (MTSLW)

(a Multi Terminal Simulator on a Linux to evaluate Web Sites)

国立茨城工業高等専門学校

信太晃司，伊藤剛志，

梅原和芳，三條光輝，渡邊高明，

杉村 康

1. はじめに

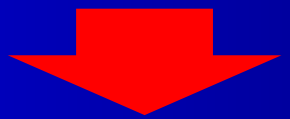
1.1 現状のサーバの処理能力評価における問題点

ベンチマークツール:

同一の要求を繰返し行うだけで、実際のシステムにおけるトランザクションの状態を反映していない.

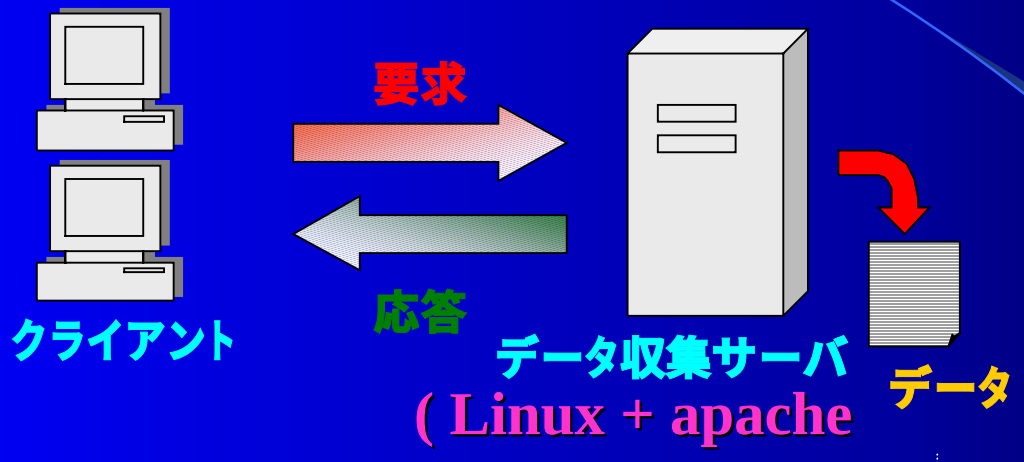
既存多端末シミュレータ[1]:

実際の要求に近いが，オープンソースではない.
標準化がされていない「通信プロトコルの下位層」でデータを収集するので，汎用性が無い.

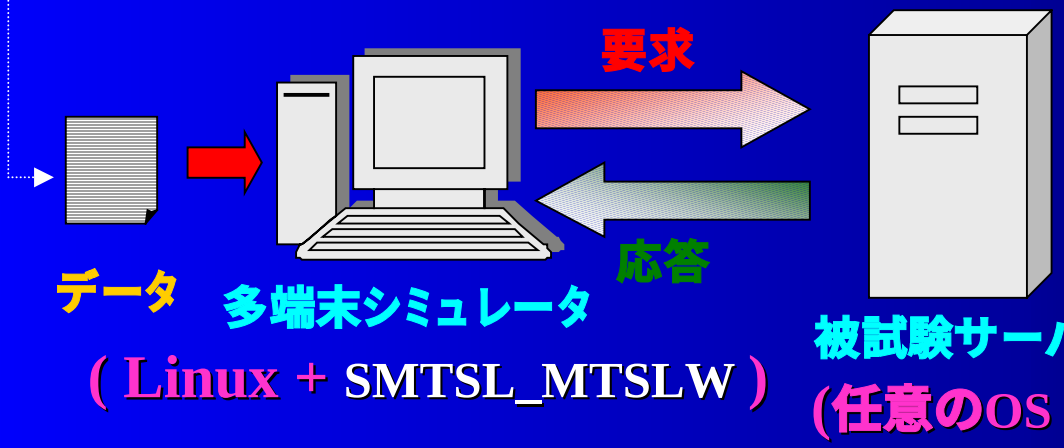


フリーの多端末シミュレータをつくろう!

1.2 Webサイト評価のために多端末シミュレータ (MTSLW) の構成 (1/2)



Data Collecting Program (SDCP_MTSLW)



Multi Terminal Simulator (SMTSL_MTSLW)

1.3 多端末シミュレータの構成 (2/2)

- 多端末シミュレータのための情報収集プログラム

  SDCP_MTSLW:

  (Sugimura Lab's Data Collecting Program for MTS on a Linux ; 以下SDCPと略記.)

- 多端末シミュレータのための負荷発生器

  SMTSL_MTSLW:

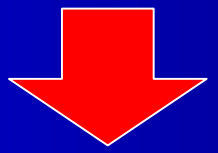
  (Sugimura Lab's MTS for a Linux; 以下SMTSLと略記.)

2. 情報収集プログラムの作成

Webサーバ(apache) からの情報収集を行うには…

2.1 Apacheの解析

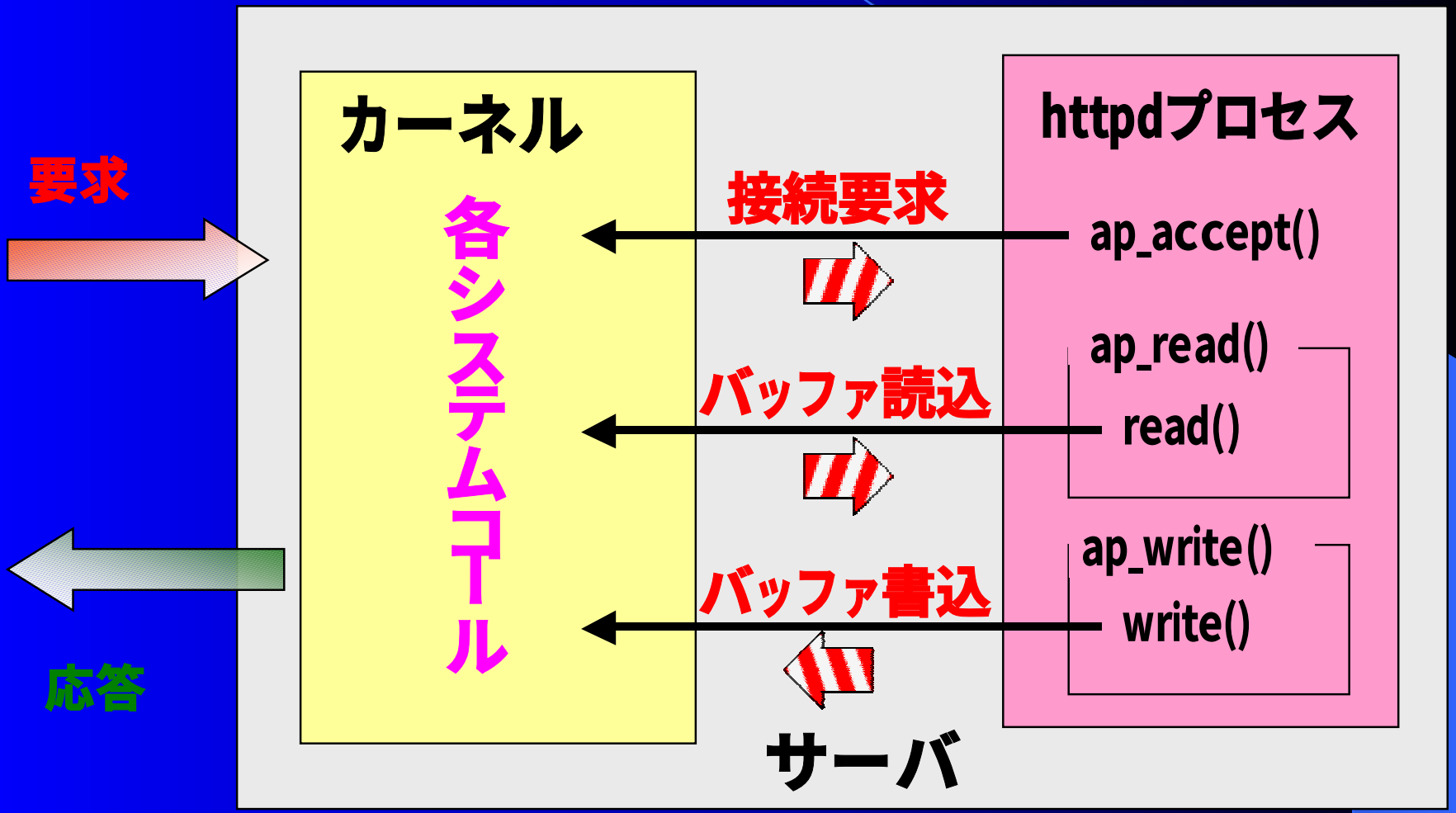
ソースコードの全行把握・・・実質的に不可能



STDB (A Super Tracer and an Analyzer for analyzing Detailed Behavior of a Linux on a Pentium family Processor[8]) による

Apacheのトレース結果からデータの
取得場所を特定する.

2.2 トレース結果


要求
カーネル
各システムコール
httpdプロセス
接続要求
ap_accept()
ap_read()
バッファ読込
read()
ap_write()
バッファ書込
write()
応答
サーバ

2.3 データの取得場所

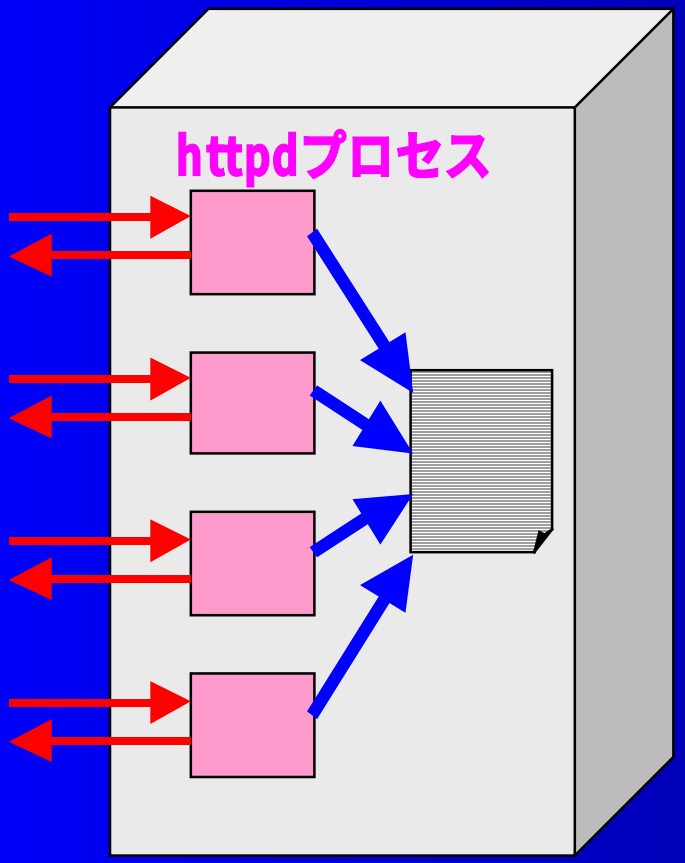
ap_accept( ) ・・・ ソケット接続の情報

ap_read( ) ・・・ 受信データの情報
（バッファサイズ4KB）

ap_write( ) ・・・ 送信データの情報
（バッファサイズ32KB）

3．ファイル共有管理機構

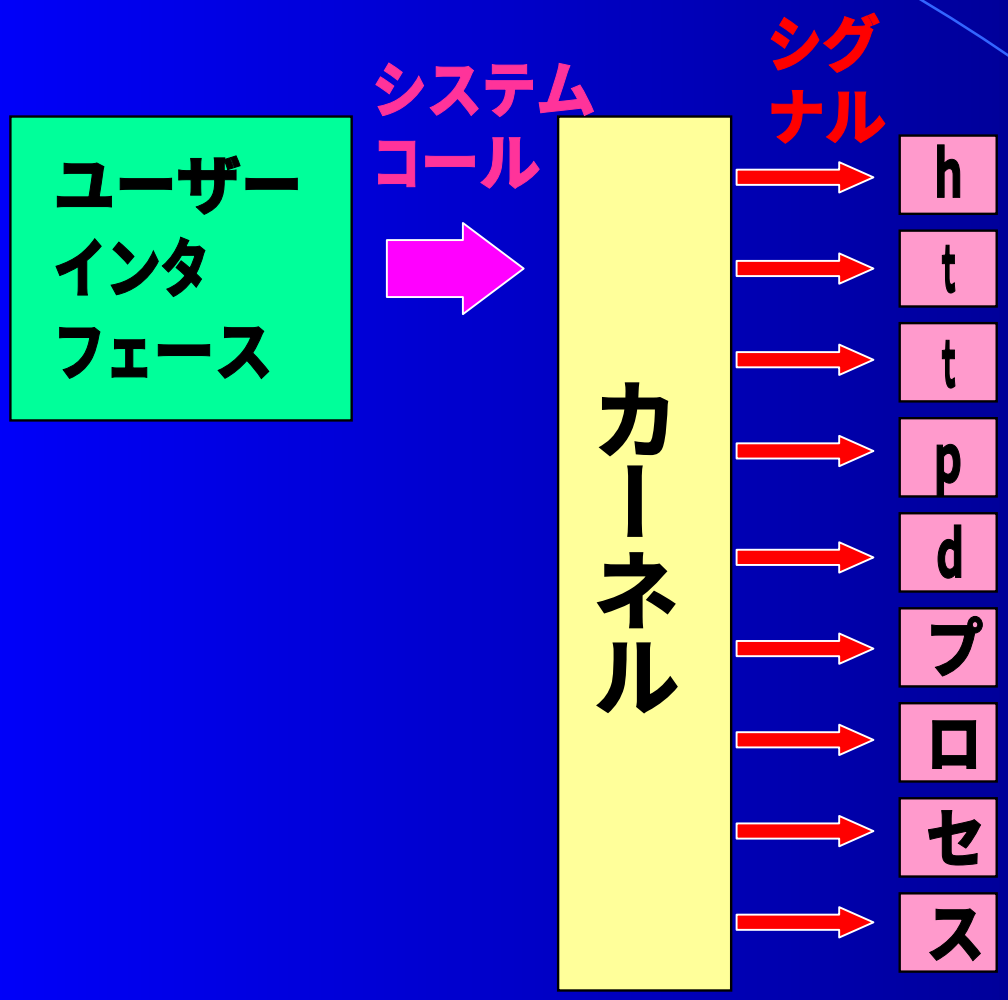
3.1 ファイル共有管理機構の必要性



Webサーバは複数のプロセスで，通常24時間無停止で動作している．

Webサービス提供中に，一つのファイルを矛盾なく管理するための管理機構が必要．

3.2 ファイル管理の実現方法



ユーザーインタフェース
からシステムコールする.

カーネルは各httpdプロ
セスにシグナルを発行
する.

httpdのシグナルハンド
ラはファイルオープン、
クローズ、チェンジの各々
の処理を行う.

3.3 ユーザインタフェース

- sdcp_open　ファイルのオープンを含むデータ収集の開始
- sdcp_close　ファイルのクローズを含むデータ収集の終了
- sdcp_change　ファイルのクローズ、オープンを含むデータ収集ファイルの切り替え

ファイル名は作成時のタイムスタンプを用いてsdcp_yymmddhhmmssとする.

3.4 システムコール

新しいシステムコールの作成

現在開発中のLinux との重複を避ける為に、新規にシステムコールを用意するのではなくstdbのシステムコール（番号は２５５番）に機能を追加した.

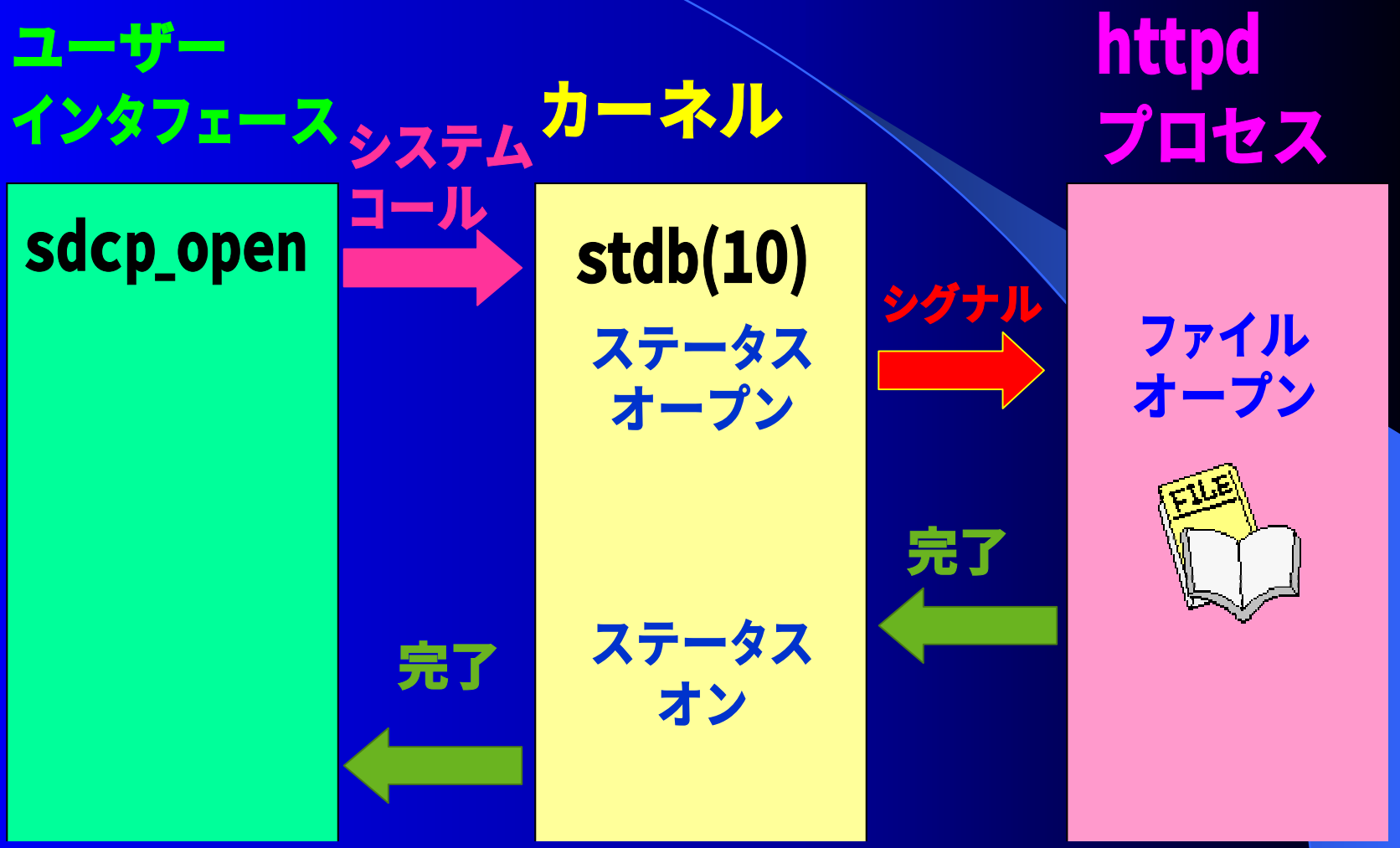
- stdb（10）～stdb(12)の処理

  ステータスを設定

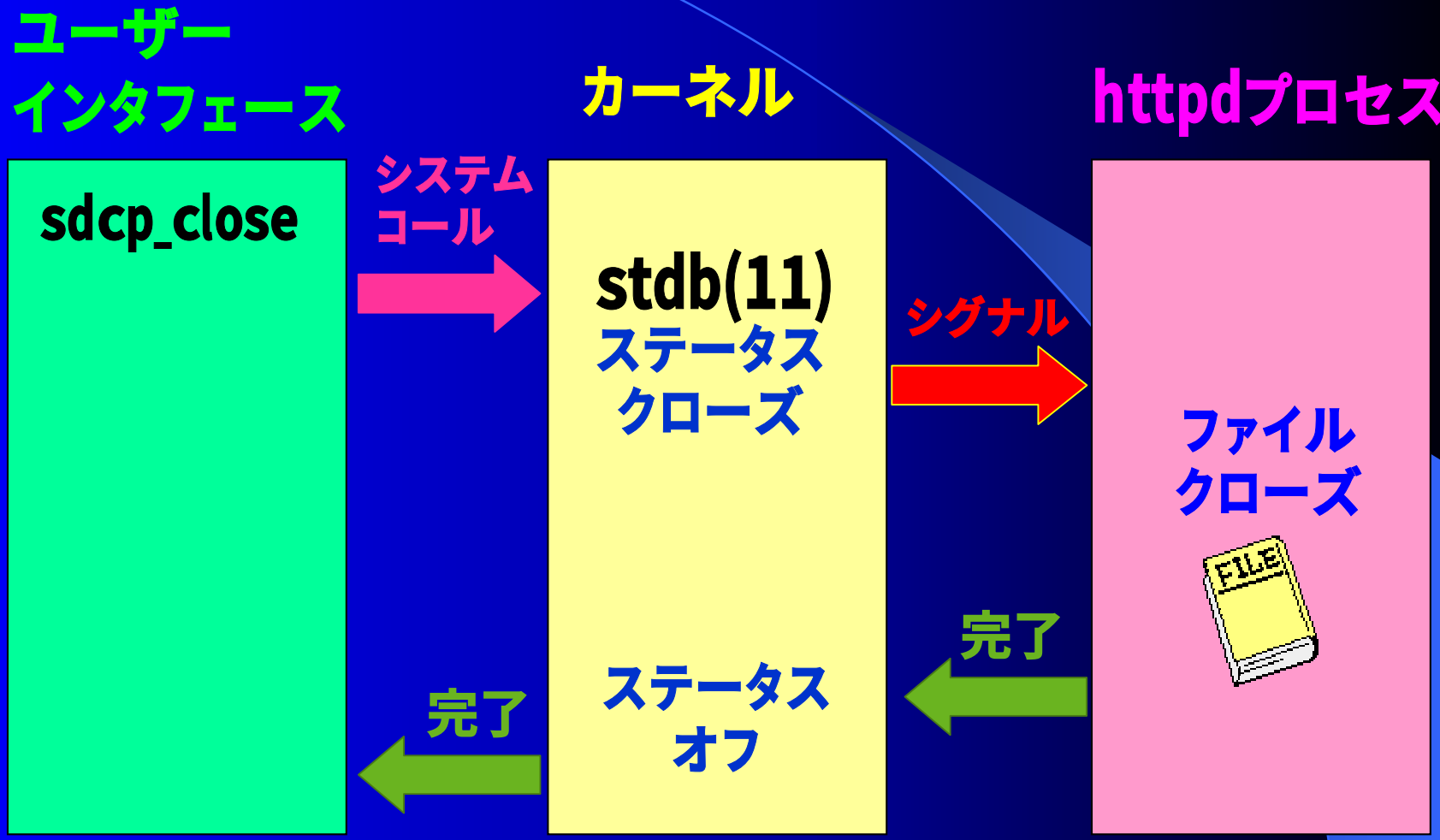
  各httpdプロセスにシグナルを発行

- stdb(10)　ファイルオープンを含むSDCPの開始
- stdb(11)　ファイルクローズを含むSDCPの終了
- stdb(12)　ファイルのクローズ･オープンを含む呼
  データファイルの切り替え

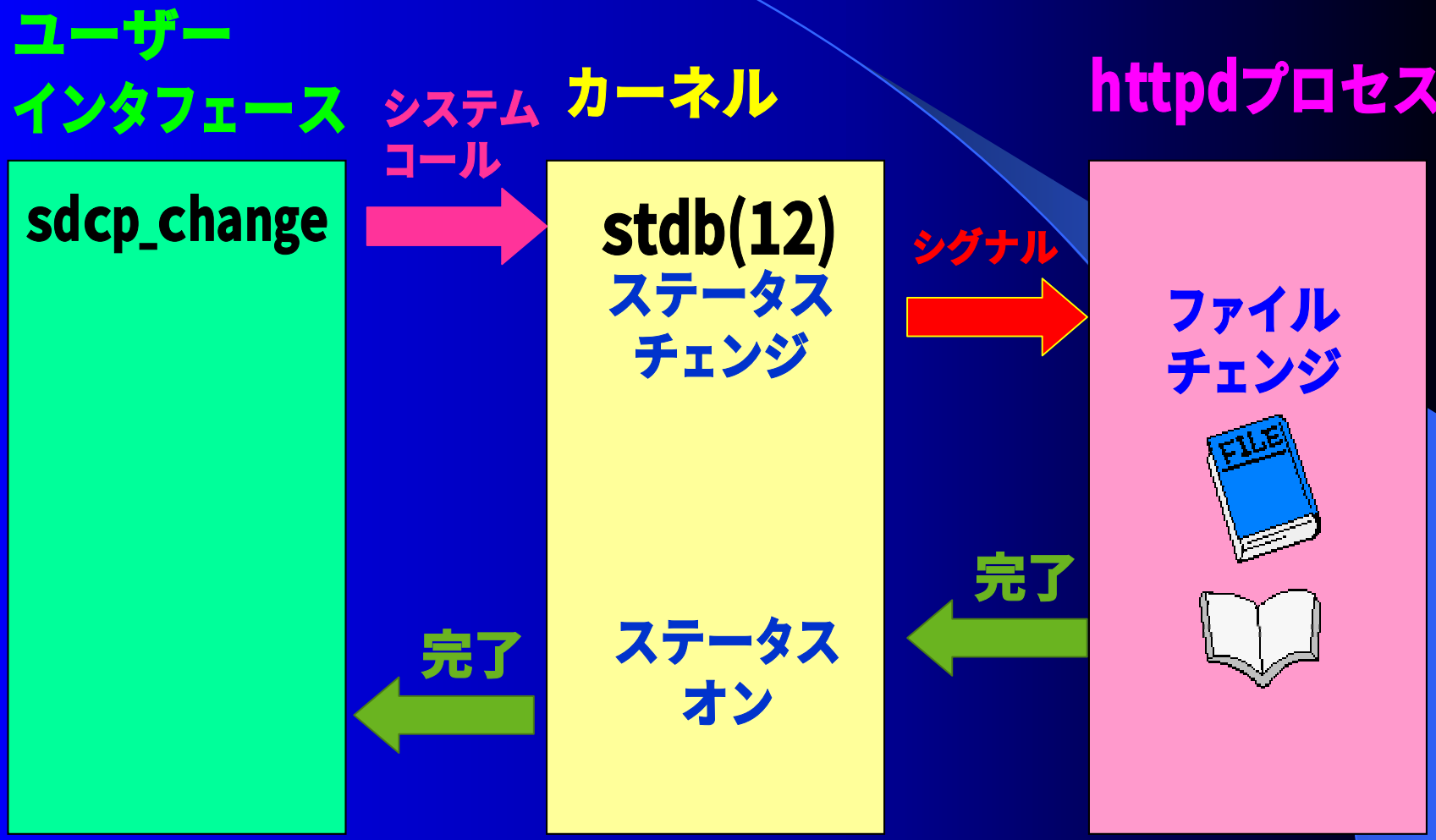
3.6 オープン処理


ユーザー
インタフェース
システム
コール
カーネル
httpd
プロセス
sdcp_open
stdb(10)
ステータス
オープン
シグナル
ファイル
オープン
FILE
完了
ステータス
オン
完了

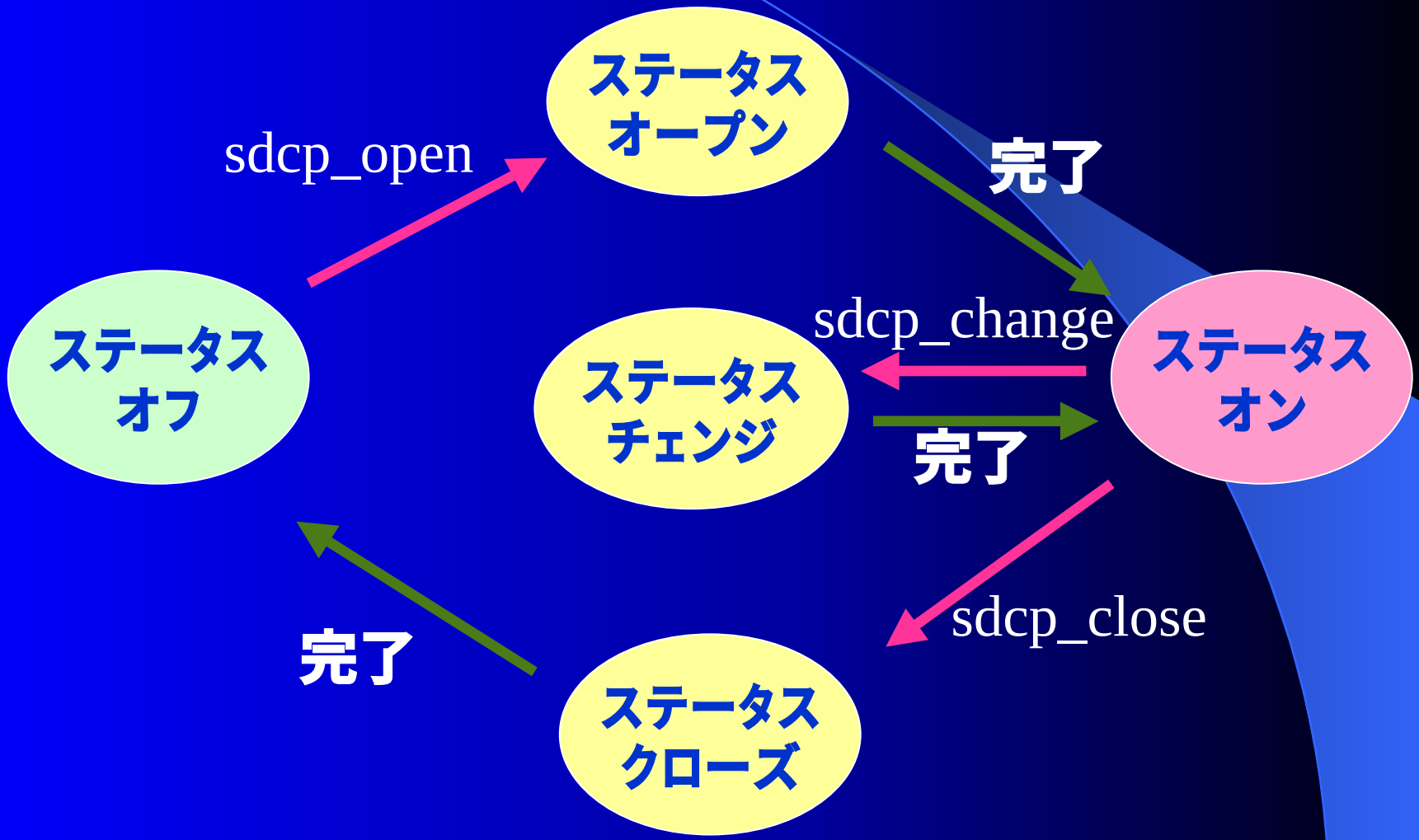
3.7 クローズ処理


ユーザー
インタフェース
カーネル
httpdプロセス
sdcp_close
システム
コール
stdb(11)
ステータス
クローズ
シグナル
ファイル
クローズ
FILE
完了
ステータス
オフ
完了

3.8 チェンジ処理


ユーザー
インタフェース
システム
コール
カーネル
httpdプロセス
sdcp_change
stdb(12)
ステータス
チェンジ
シグナル
ファイル
チェンジ
FILE
完了
ステータス
オン
完了

3.9 状態遷移図


ステータス
オープン
sdcp_open
完了
ステータス
オフ
sdcp_change
ステータス
チェンジ
ステータス
オン
完了
sdcp_close
完了
ステータス
クローズ

3.10 共有ファイルへの出力

各httpdプロセスは上記で作成した各々のファイルポインタを保持し、それを用いてそれぞれのデータ取得場所でファイルへの書き込みを行う。書き込みが行われたなら、フラッシュを行いストリームの同期をとる.

このようにして,各プロセス間で矛盾なくファイルを用いることのできる管理機構を実現.

4．出力結果

4.1 出力項目と出力例 (1/3)

- ap_accept関数

  クライアントIPアドレス

  応答したhttpdのプロセスID番号

  取得した時間（10 [ms] 単位）

```
Accept IP address: 172.16.53.242、pid = 563、
                                jiffies = 25318 * 10 [ms]
```

4.2 出力項目と出力例(2/3)

- ap_read関数

  応答したhttpdのプロセスID番号
  取得した時間(10[ms]単位)
  受信データ

```
### The buffer that passed ap_read ###
pid = 563、 jiffies = 25318 * 10 [ms]

GET /~umeh/index.html HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.72 [en] (X11; U; Linux 2.2.16s i686)
Host: srroumeh
Accept: image/gif、 image/x-xbitmap、.....( 後省略. )
```

umeh配下index.htmlを取得しようとしている.

4.3 出力項目と出力例(3/3)

- ap_write関数

  応答したhttpdの プロセスID番号

  取得した時間 （ 10 [ms] 単位 ）

  送信データ (Httpヘッダ, Html文書)

```
### The buffer that passed ap_write ###
pid = 563 、 jiffies = 25318 * 10 [ms]

HTTP/1.1 200 OK
Date: Thu、 07 Feb 2002 06:19:04 GMT
Server: Apache/1.3.12 (Unix)  (Red Hat/Linux) PHP/….  (中略)
<html>
<body bgcolor=blue>
<img src = “image1.png”> ….(後略)
```

5. SDCPの正常性確認

SDCPにおける呼データの収集漏れが無いかをチェックするため, その試験においては, tcpdumpを起動し, 得られた16進データを変換したものと, 収集されたデータとの比較により, 正常性を確認した.

6. 負荷発生器 (SMTSL) について

6.1 ab流用の検討

負荷発生器 (SMTSL) の検討においては, 先ず, Linuxの負荷発生用プログラム“ apache HTTP server benchmarking tool (ab) ” を流用できないかを, そのソースファイルより解析･検討を行なった.

その結果,次の特徴があることが判った.

6.2 abの特徴

- 1種類のweb アクセスを何度も繰り返す.
- 起動時のkeepalive指定に関わらず,webサーバ側からkeepaliveがくれば回線断を行わない.
- keepaliveが来なかった場合,writeとreadの一組の後,サーバ側からの回線断を契機に,接続をやり直す.
- プロセスは一つしか使用していない.
- readとwriteはselectによって制御されているので並行動作が行われているが,connectやclose中にはreadやwriteのイベントがあっても処理されない.

6.3 実現要事項

SMTSLにおいては，被評価システムに高負荷を与えることを実現するために，ab内に独立したSMTSL等を追加することにより，以下を実現.

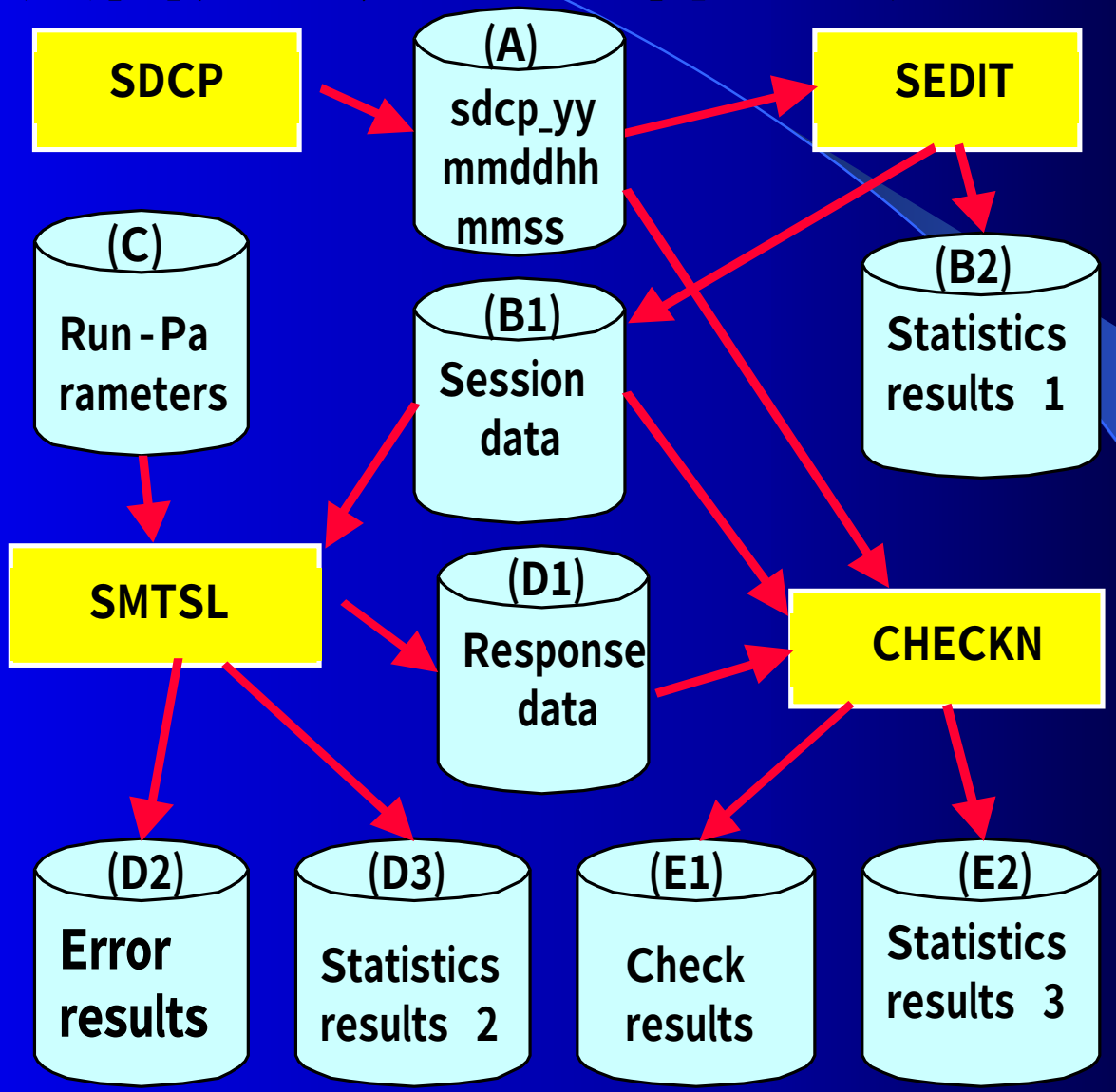
- プロセスの複数化
- 送受信データの極小化
- 高負荷モード及び正常性監視モードの 二つの起動モードの実現
- 適切な終了契機の実装
- 適切な補助プログラムの追加実装

6.4 SMTSLと,その補助プログラム

SMTSLと, その補助プログラムは, 以下の3つのプログラムから構成される.

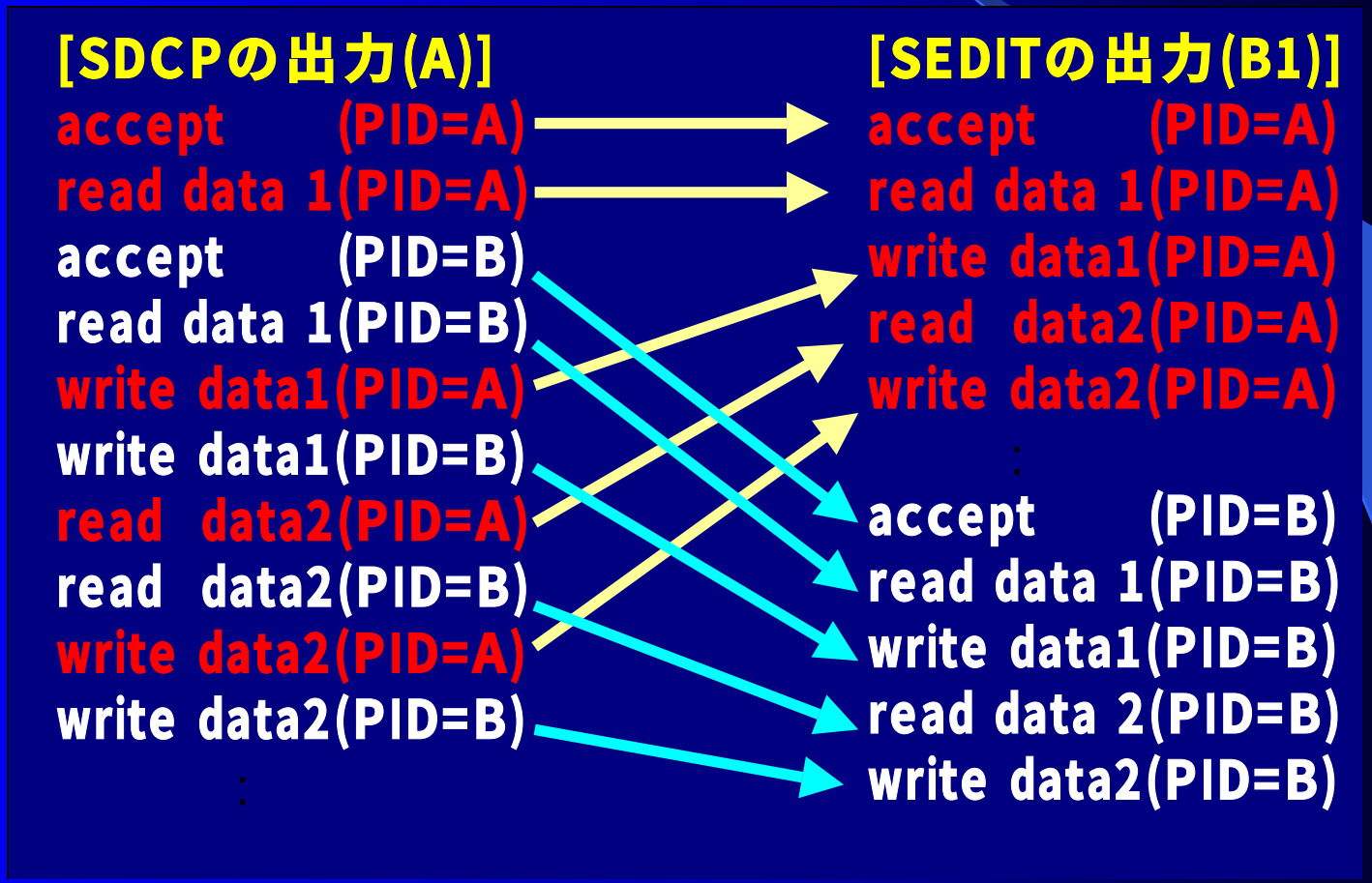
- SDCPデータ変換プログラム（SEDIT）
- 負荷発生プログラム（SMTSL）
- 正常性チェックプログラム（CHECKN）

6.5 入出力ファイルと各プログラムとの関係



6.6 SDCPデータ変換プログラム（SEDIT）の機能

SDCPが出力した呼データファイル内容を編集してセション毎にまとめた セションデータファイルを生成.



6.7 負荷発生プログラム(SMTSL)の機能

- 起動パラメータとセションデータファイルの読込み
- 起動パラメータで指定された数の子プロセスを生成
- 被試験システムへの要求として複数の呼を並行して送出
- 応答データ長のチェックとエラー情報のファイル出力
- 運転停止状態検出時におけるプログラム停止処理
  (運転停止条件は予め起動パラメータ中で指定)
- 応答データにセション番号とセション内番号の付加
  (正常性監視モードのみ)

List 5　A list of run parameter of the SMTSL.

1. 運転モード {高負荷モード or 正常性確認モード}
2. 被測定システムのIPアドレス
3. 被測定システムのポート番号
4. 同時接続数（子プロセス数）
5. セション間隔［ 秒 ］（ デフォルトは零 ）
6. 運転停止条件:
   (A) 全セションデータを処理した時点
   (B) 指定トランザクション数のレスポンス時点
   (C) データ長エラーが指定数発生した時点

List 6 Statistics data.

1. SMTSLの起動パラメータの全て
2. 呼発生開始時間
3. 運転終了時間
4. 全セッション数
5. セッション当り平均トランザクション数
6. 単位時間当たりのトランザクション数
7. 単位時間当たりの全/要求/応答の平均データ転送量[Mbps]
8. 平均/最大/最小応答時間
9. 1トランザクション当たりの全/要求/応答/応答のContents
   の平均，最小，最大転送データ長［ バイト ］
10. 応答データ長不一致数
11. 応答データ長不一致率[%]

6.8 正常性チェックプログラム（CHECKN)の機能

- 以下のファイルの入力
  1) SDCPが出力した呼データファイル
  2) SEDITが出力したセションデータ
  3) SMTSLが出力した応答データ
- SMTSLの受信データとSDCPのwriteデータのContentsの比較を行い, 相違を検出した場合, 該セション番号, 該セション内番号, SMTSLの受信データのContents, 及び, SDCPのwriteデータのContentsのファイルへの出力
- 統計結果のファイルへの出力

7. むすび

以上の手法により，Pentiumを使用したRed Hat Linux 6.2を例にとって，処理能力評価を実際の呼データを使用して客観的に評価するシステムMTS LWが実現可能であることを示した.

尚，当研究は，現在進行中であり，将来の同Linux 7.3での実証，並びに，GPLに基づき，以下のURLでのソースプログラムやマニュアル等の今年度末公開を目指している.

http://www.ibaraki-ct.ac.jp/ece/yas/mtslw/index.html

謝辞

以下の方々に, 深く感謝致します.

素晴らしいLinuxを実現してくださったLinus Torvalds様と仲間の皆様,

Apacheを開発されたApache Groupの皆様,

並びに, GPLの元で種々のプログラムを提供してくださった皆様.

「完」